Using Taint Tracking to Improve Energy Efficiency of Always-on Smartphone Apps

Haichen Shen, Aruna Balasubramanian, Anthony LaMarca*, and David Wetherall
University of Washington, *Intel

1. MOTIVATION

Sensor monitoring apps on smartphones provide a variety of services, ranging muting the ringer in certain locations, to evaluating Parkinson patients, to encouraging users to stay off of junk food. Unfortunately, sensor apps often require continuous monitoring, and are heavy drain on the mobile's battery. The problem is that the sensors are on the same controller as the CPU and taking a sensor reading for a few microseconds results in the CPU being awake for a few hundreds milliseconds.

To address this problem, researchers have proposed a sensor hub architecture, where the sensor data is collected and processed on an alway-on micro-controller, distinct from the main processor [2]. While these systems have been demonstrated to be effective for simple sensor applications such as a pedestrian step counter, it is unclear how much savings a sensor hub can offer a complex sensor application. The main problem is, it is unclear how applications use sensor data. If applications buffer sensor readings and process them periodically, they can easily be partitioned to the sensor hub. However, if they process sensor data as they are read, then a sensor hub may worsen the performance. Without understanding sensor usage, it is difficult to design a general sensor hub architecture.

We propose that *taint tracking* is a useful tool in addressing this issue. By tainting sensor data as it is read, we can track it through a running system and determine when sensed data produces user-observable events. We then can determine which sensor data contributed and how old it was. Moreover, we can look at the types of algorithmic transformation the data underwent. In this way, we can, without performing a static code analysis, determine how applications make use of sensor data and whether they would be good candidates for optimization via a sensor hub.

2. TAINT TRACKING

We use *taint tracking* to systematically track data from the sensor to a user-perceivable action by the application. We adapt a system called TaintDroid [1] to perform this tracking. TaintDroid is a privacy monitoring system that tracks when private information is being leaked over the network. Our goal is different, and we make several modifications to TaintDroid accordingly. The most important modification is in tracking taints through control flows; TaintDroid only tracks information leak through data flow. In our modified TaintDroid we tag variables that has been tainted with *sensor data*. We provide a unique tag for each sensor data; for example, the 2nd accelerometer reading has a different tag than the 3rd. We also log when the tainted variable is used in a user-perceivable manner. In our implementation, we say a tainted variable is user-visible if it is sent over the network, written to a file, or displayed on the screen. As a result, our system allows us to track when the sensor data is being collected (based on tainting a variable), and when sensor data produces a visible side-effect (based on the log).

To add control flow tainting, we identifying potential taint blocks, for example, an *if* statement controlled on a tainted value. We then track the flow of sensor data through the taint block. Tainting control flow is essential for sensing applications, because these applications often perform computation based on the value of the sensor reading; for example, based on whether the sensor reading is greater than a threshold value. However, the computation itself may not use the sensor value, making data flow taints alone insufficient.

Figure 1 show the results of our taint analysis on an off-the-shelf pedometer app. As an example reading, the point (30, 0.8) shows that 20% of the time, the app only uses 1 in 30 contagious sensor readings in a user-perceivable manner. If the 30 contiguous readings were buffered for those 20% cases, the application could have reduced energy consumption by 95% for those cases, without affecting semantics.

3. NEXT STEPS

We are currently analyzing the top off-the-shelf sensor applications using our tool. The goal of our analysis is to understand the energy implications of different sensor hub tasks, for example, sensor data buffering, simple sensor thresholding, etc, for a given application, under varying

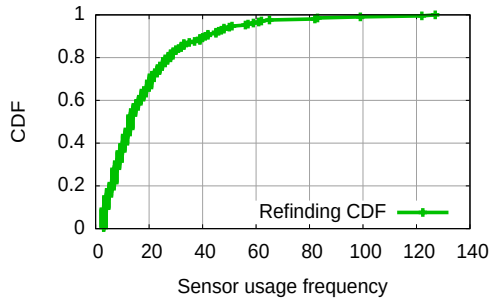

Figure 1: The sensor usage of a pedometer app.

scenarios. We believe that taint tracking is not only a powerful tool to understand sensor usage in applications, but it can inform the design of a system architecture that leverages sensor hubs to improve energy efficiency.

4. REFERENCES

[1] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI 2010*, 2010.

[2] B. Priyantha, D. Lymberopoulos, and J. Liu. Littlerock: Enabling energy-efficient continuous sensing on mobile phones. In *IEEE Pervasive*, 2011.